PATENT
ATTORNEY'S DOCKET NO. 032001-011

"Express Mail" Mailing Label No. EL 521 508 415 US
Date of Deposit October 3, 2000

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above, with sufficient postage affixed, and is addressed to BOX PATENT APPLICATION, Assistant Commissioner for Patents, Washington, D.C. 20231.

Seini Matangii

(Typed or printed name of person mailing paper or fee)

HIERARCHICAL STORAGE ARCHITECTURE FOR RECONFIGURABLE LOGIC CONFIGURATIONS

ABSTRACT OF THE DISCLOSURE

1. Field of the Invention

The present invention relates to reconfigurable computing.

2. State of the Art

A Field Programmable Gate Array (FPGA) is a single-chip combination of computing elements and storage elements. The computing elements can be configured to implement different logic functions depending on the values stored in the storage elements. A collection of such values that can configure all the computing elements on the chip will be referred to as a "configuration plane". A collection of values that is a subset of a plane will be referred to as a "configuration."

In a conventional FPGA, there is only enough on-chip storage for a single configuration plane. In a variant of FPGAs known as Reconfigurable Logic, there may be enough on-chip storage for multiple configuration planes. In reconfigurable logic there is typically some mechanism for rapidly changing which plane is currently configuring the computing elements. In addition, there is typically some mechanism for loading the multiple planes from off-chip storage, which can result

in virtually unlimited configurations for the chip. However, the time required to load the off-chip configuration data is the bottleneck for current implementations.

The off-chip loading is typically handled by either a caching or a pre-fetch strategy. In a caching strategy, an on-chip cache of the most recently used configurations is stored, and in the event of a cache miss, the chip is stalled until the configuration can be loaded from off-chip. This is a delay of several hundreds of clock cycles for the current generation of reconfigurable logic. In a pre-fetch strategy, the overall schedule of configuration invocations is analyzed and the appropriate configurations are loaded into the configuration planes before they are needed, ideally avoiding stalling the chip. However, the more time required to load an off-chip configuration, the more branching in the configuration schedule will be encountered between the pre-fetch and the actual use, possibly invalidating the original pre-fetch decision and stalling the chip.

SUMMARY OF THE INVENTION

The present invention, generally speaking, provides a hierarchy of configuration storage. The highest level of the hierarchy is an active configuration store; the lowest level is an off-chip configuration store; in between are one or more levels of configuration stores. Every configuration is promoted from the lowest off-chip level, through each level, up to the highest active level. Each ascending level of the hierarchy has a decreasing latency time required to promote a configuration to the next higher level of the hierarchy, and a decreasing amount of available storage. This separation into levels allows the amount of available storage to be adjusted depending on the inherent latency of the level's storage mechanism, where a longer latency requires a larger cache. This in turn allows the total required storage for a given performance level to be minimized.

BRIEF DESCRIPTION OF THE DRAWING

The present invention may be further understood from the following description in conjunction with the appended drawing. In the drawing:

Figure 1 is a diagram of an exemplary configuration storage hierarchy;

Figure 2a is a simplified example of a configuration to be compressed;

Figure 2b is a compressed format used to represent the configuration of Figure 2a, the bits of the representation being further compressed;

Figure 3 is a diagram showing an example of a suitable on-chip cache;

Figure 4 is a diagram showing an example of decompression;

Figure 5 is a diagram showing an example a of planes/configuration table;

Figure 6a is a block diagram of a portion of a memory plane stack;

Figure 6b is a diagram of a group of corresponding memory cells, one cell form each plane of the memory stack of Figure 6a;

Figure 6c is a diagram of an alternative embodiment of the memory stack of Figure 6a in which separate "function" and "wire" stacks are provided;

Figure 6d is a diagram of separate memory stacks provided for control, datapath and memory configuration, respectively;

Figure 6e is a diagram of a common memory stack provided for control, datapath and memory configuration; and

Figure 7 is a schematic diagram of an alternative embodiment for a single bit of the memory stack of Figure 6a.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Referring now to Figure 1, in a preferred embodiment of the present invention, an FPGA or reconfigurable logic device is provided with a configuration storage hierarchy having multiple levels, e.g., four levels: 1) off-chip storage, 2) compressed cache storage, 3) decompressed configuration planes, and 4) one or more active configuration planes. A description of each level follows, proceeding from lowest to highest level.

Off-Chip

The off-chip level of storage may be implemented in a variety of technologies, including without limitation EEPROM, RAM, hard drive, or I/O port. Preferably, the external storage device is memory mapped (corresponds to address entry in system CPU memory access space), and an instruction to load a specific configuration from off-chip storage device will include the configuration's starting address and length. The length of a configuration will vary depending on: how many computing elements it configures, the specific function for each computing element, and the amount of compression achieved.

A configuration may include an arbitrary number of computing and/or routing elements. Nor is there any restriction that the elements be contiguous on the chip. Partial reconfiguration may be used to support a "data-in-place" computing style where some computing elements configured as registers and holding active data are left untouched, while other computing elements are reconfigured to perform new functions on the data. Partial reconfiguration may be performed. For "data in place," storage contents are left in place at either/or register and local memory elements. The control logic or wiring interconnectivity can be updated with new certification data while the rest of the configuration data fields for the storage remains unchanged. In a preferred embodiment, routing between elements can remain static while the control codes are updated. In both of these cases, selected

subsets of configurations are used, resulting in effective benefits of partial reconfiguration.

The off-chip configurations are stored in a compressed format. One possible compression scheme is described here. Referring to Figure 2a, the computing elements on the chip are in a two dimensional X and Y array. A computing element is configured by storing an opcode (e.g., 1, 3, 7, etc.) in the computing element. Routing elements occupy rows and columns where all elements in a row have the same Y coordinate, and all elements in a column have the same X coordinate. Referring to Figure 2b, a single configuration consists of a series of instructions, to be executed in sequence, all with the following three-field format: Y control, X control, opcode. The Y control is a binary number from 0 to N-1, where N is the maximum possible Y coordinate. The X control is an N bit wide word, where N is the maximum possible X coordinate. In other words, the X control has one bit for every column, and the Y control is decoded for each row. In the row enabled by the Y control, for each element in the row where the corresponding X control bit is a 1, the specified opcode will be loaded into the element. On top of this "common configuration" compression, the entire configuration (sequence of instructions) may be bit-wise run length compressed.

In an alternative implementation, the Y control may not be encoded if the savings from simultaneously loading multiple rows with the same opcode outweighs the savings from encoding the row coordinate.

In addition to being compressed, the configurations may also be encrypted.

The number of bits used to configure a single element may vary. It is possible to apply, for example, Huffman encoding to the set of possible configuration codes so that the more frequently used codes require fewer bits than the less frequently used codes. Even if a fixed bit-width is used for the opcode, maximizing the number of leading zeros will help in a run length compression scheme.

Compressed Cache

The on-chip compressed cache can be loaded directly from the off-chip configurations. The on-chip cache has its own dedicated DMA server. The configurations are loaded directly from off-chip without any modification, in compressed format. As a result, more configurations can be stored in a given amount of cache, and the off-chip loading time is minimized.

Referring to Figure 3, one possible implementation of the cache is as follows. The on-chip compressed cache may be implemented as a RAM with multiple cache "lines", where each line consists of a configuration field, a contents addressable field, and a tagged bit field. The contents addressable field will store the address of the configuration, which is the same as the off-chip address used to load the configuration. The tagged bit field is used during a search of the cache for a given configuration. The tag bit is set to TRUE for any line with an address field that exactly matches a searched for address, and is set to FALSE otherwise. Whenever a configuration is loaded into the cache, a search is performed first to check if there is already a line with the same address. If so, the off-chip configuration is loaded on top of the existing line in the cache. If not, the first available line is used. A separate counter with wrap around is maintained to indicate the first available line. If the first available line's address field is not equal to zero, an error flag is raised. When a line in the cache is freed, its address field is set to zero. Instead of a wrap around counter, an alternate method for identifying an available line is to search for a zero address and use the first available.

Decompressed Planes

The decompressed planes are loaded with configurations from the compressed cache, with stream-oriented decompression and decoding. Once they are in the decompressed planes, configurations can be moved into the active plane in

as little as a single clock cycle. The decompressed planes serve as the rapid staging area for the active plane.

Referring to Figure 4, one possible implementation of the decompression and decoding process is as follows. A fixed bit-width is assumed for the length field of the run-length compressed bitstream. The length field value is loaded into a count down counter. The next bit is shifted into a shift register until the counter reaches zero or the register is filled. The bit-width of the register corresponds to the length of a single configuration instruction. The instruction's X, Y, and opcode fields will have been zero-filled so that the fields are always the same bit-width. When the register is filled, the fields will drive the loading of the decompressed plane accordingly. The process continues until a length field of zero is encountered.

If the configuration instructions are encrypted, they will be decrypted after each configuration instruction is decompressed. In this case, local hardware would intervene to perform the decryption before the disbursement in the configurable storage planes.

Referring to Figure 5, a separate table is maintained that stores the address of the configuration that is currently loaded in each decompressed plane. While the chip is executing, this table can be used to verify that the intended configurations have actually been pre-fetched and are still resident in the planes. This table can also be used to save and restore the state of the chip in the event of an interrupt. This table can also be used to boot some initial configurations into the chip during power-up.

Active Plane

The active plane can be loaded from any of the decompressed planes. A particular embodiment of a memory plane stack 1200 is shown in Figure 6a. In the illustrated example, the top two planes 1206, 1205 of the memory plane stack are configuration planes. Configuration data stored in these planes is applied to the

reconfigurable logic. In the illustrated embodiment, "function" configuration data and "wire" configuration data is stored in different planes. The bottom memory plane 1200a provides external access to the memory stack. Intermediate planes function, for example, as a configuration stack, storing configurations expected to be used by not presently active. In an exemplary embodiment, memory plane 0 is single port, for single-channel read and write between system memory and configuration storage. The remaining memory planes are dual port, having one read port and one write port. Dual port supports simultaneous loading and recirculation of configuration data with the local "stack." If no data compression is used, then simultaneous real-time monitoring is possible, e.g., by writing out a "snapshot" of one or more planes of the stack.

A group of corresponding memory cells, one cell from each plane of the memory stack, is shown in Figure 6b. The ports of all of the cells are interconnected so as to allow an operation in which the contents of a cell within any plane may be read and then written to the corresponding cell of any other plane. For example, by activating the appropriate control signal, the contents of plane 4 may be read and written into plane 6. Such an operation may be accomplished, preferably, in a single clock cycle, or at most a few clock cycles. Configuration data is loaded from external main memory into plane 0 of the memory stack in anticipation of its being transferred into a configuration plane.

Alternatively, separate "function" and "wire" stacks may be provided, as shown in Figure 6c. Using this arrangement, function and wire configurations may be changed simultaneously. Similarly, configuration stacks for configuration of control, datapath and memory may be combined (Figure 6d) or separate (Figure 6e).

A schematic diagram of an alternative embodiment of a cell stack is shown in Figure 7, showing a cross section of several configuration planes 1301-1304 and the lockable fabric-definition cell 1305 that produces a Fabric_Define_Data bit for a

single bit location. These bits are aggregated in order to form sufficient bit numbers for functional cell type definition. For instance, a four bit grouping might designate between four to sixteen different cell type definitions. The other latch sites below the storage cell are for additional configuration plane data available for swapping as needed by functional scheduling requirements. These storage locations can be written and read to from a common configuration data bus structure. The Config_Read_Data and Config_Load_Data buses 1307 and 1309, although shown as being separate, can be combined as a single bi-directional bus for wiring efficiency. This bus structure allows configuration data to be written as needed. The Swap_Read_Plane buffer 1311 allows existing configuration plane data contents to be swapped among differing configuration planes on a selectable basis. For instance, the current operation plane of data can be loaded from configuration plane 1 to configuration plane 2 by the use of the Swap_Read_Plane buffer 1311. The structure shown in Figure 7 is similar to a conventional SRAM memory structure which allows a dense VLSI circuitry implementation using standard memory compiler technology. This structure could also be implemented as a conventional dual port RAM structure (not shown) which would allow for concurrent operation of the write and read data operations. Unlike Figure 6b, the example of Figure 7 assumes separate configuration stacks for each configuration plane as described hereinafter. That is, the bit stack produces only a single Fabric_Define_Data bit instead of multiple fabric definition data bits as in Figure 6b. The bits could also be extended to include registers operating in a like fashion.

If the Data_Recirc_Read line 1313 is also connected to data storage locations that are used for normal circuit register operation, then real time monitoring of device operations can be utilized by the operating system for applications such as RMON in internetworking application area or for real time debug capability. The RMON application basically uses counter operation status from registers in order to determine system data operation flow characteristics.

It will be appreciated by those of ordinary skill in the art that the invention can be embodied in other specific forms without departing from the spirit or essential character thereof. The presently disclosed embodiments are therefore considered in all respects to be illustrative and not restrictive. The scope of the invention is indicated by the appended claims rather than the foregoing description, and all changes which come within the meaning and range of equivalents thereof are intended to be embraced therein.